**Encapsulation**

* Explain the meaning of encapsulation?

Encapsulation is a way to restrict the direct access to some components of an object, so users cannot access state values for all of the variables of a particular object. Encapsulation can be used to hide both data members and data functions or methods associated with an instantiated class or object

* Highlight a benefit of Encapsulation?

Hiding data: Users will have no idea how classes are being implemented or stored. All those users will know is that values are being passed and initialized.

More flexibility: Enables you to set variables as red or write-only. Examples include: setName(), setAge() or to set variables as write-only then you only need to omit the get methods like getName(), getAge() etc.

Easy to reuse: With encapsulation, it's easy to change and adapt to new requirements

* Provide an application of encapsulation.

Encapsulation is simply "minimizing inter-dependencies among separately-written modules by defining strict external interfaces" . That is to say that when I am building a module, I want a strict contract between my clients and me on how they can access my module. Reason being that, I can improve the inner workings without it AFFECTING my client's, life, application or whatever they are using my module for. Because their "module" does not exactly depend on the Inner workings of my module but depends on the "external interface", I made available to them.

So, if I don't provide my client with a setter and give them direct access to a variable, and I realize that I need to set some restriction on the variable before my client could use it, me changing it, could be me, changing the life of my client, or application of my client with HUGE EXPENSE. But if I provided the "strict contract" by creating a "strict external interface" i.e setter, then I can easily change my inner workings with very little or no expense to my clients.

* Use a code example encapsulation from the program wrote